**CSS - Estilos de la página - Maquetación**

Lenguaje de estilo en cascada.

en html los elementos principales son los tags

en css son las reglas (cada bloque de código de css)

están compuestas de 2 partes: Selector y bloque.

1. El Selector: lo primero que se pone. sobre qué elemento se aplican los estilos. Pueden ser tipo ‘p’, clase, id…Las reglas se ponen entre llaves en bloque, con pares propiedad seguida de dos puntos y su valor.
   1. Tag: etiqueta html como img, p, div, etc. Esto aplicará un estilo común a todos los elementos con esa tag. Es el más usado.
   2. Class: se indica con un punto ‘.’ seguido del nombre de la clase.
   3. ID: se indica con ‘#’ seguido del nombre de ID.
   4. Child: a menudo, en lugar d ebuscar elementos por clase o id (puede que no tenga), tengamos que seleccionar un elemento por su posivción en un html. En css, el selector de hijo se indica con ‘>’. la estructura sería padre > hijo:
      1. “div > img {}” nos permitiría añadir propiedades únicamente a las imageenes contenidas dentro de un div, pero no a las de fuera de estos. Dejando un espaco en blanco en vez de con el ‘>’, no coje únicamente los hijos directos, sino todo lo que esté dentro del padre que sea del tipo indicado (incluso que esté dentro de otros contenedores):
         1. .container img {}: añade propiedades a todas las imagenes dentro de la clase container, aunque estas estén dentro de un “.subcontainer”
         2. .container>img {]: añade propiedades únicamente a las img que sean hijos directos de .container. Aquellas dentro de “.subcontainer” no se modificarán, porque son nietos y no hijos.
   5. Pseudo-selectores: se aplican sobre determinadas partes de un selector que ya existe. Se indican con un dos puntos “:” después del selector.
      1. **Hover**
      2. **After**
      3. **Before**
      4. Focus
      5. active
      6. …
2. Unidades de medida (para altura, padding, tamaño borde, etc etc..)
   1. Píxeles (px): es la unidad más estricta, ya que un píxel tiene el mismo tamaño en todos lso dispositivos
   2. Porcentajes (%): es una unidad relativa en base al tamaño del padre inmediato que lo contiene.
   3. vh: es mejor que los píxeles y también que los %, porque hace referencia al tamaño de la pantalla directamente. 100vh es el 100% de la pantalla del dispositivo, 50vh la mitad de la pantalla, etc..
      1. Se usa para las cosas más grandes. Se puede indicar al principio para el contenedor principal, de manera que ocupa la pantalla entera, y los hijos ya con %.
3. Prioridades:
   1. Tendrán más prioridad las reglas que estén más abajo
   2. Especificidad: Si la regla es más específica, tiene más prioridad y se mantiene, aunque haya después una regla general más abajo.
      1. **UNA REGLA DE CSS ES MEJOR COMO MÁS ESPECÍFICA ES**
   3. !important: solo como último recurso. Sobreescribe cualquier otra regla y se aplica la indicada, aunque por prioridad no tuviera que aplicarse. Se indica después de la propiedad, pero antes del ‘;’ del final de la línea.
      1. Cuando ya tenemos importants en el código, vuelven a aplicarse el resto (una con important mñas abajo que otra con important, tendría prioridad, pero una más específica con important volvería a ser más prioritaria que otra aunque estuviera definida antes)

(leer artículo metodología [BEM](https://webdesign.tutsplus.com/es/articles/an-introduction-to-the-bem-methodology--cms-19403) y este [también](https://medium.com/@dannyhuang_75970/what-is-bem-and-why-you-should-use-it-in-your-project-ab37c6d10b79))

Mirar la web de MEDIUM, que tiene cosas útiles para seguir actualizados (sección technology, hay cosas frontend, etc)

**POSICIONAMIENTO EN CSS**

1. Display: indica cómo se posiciona un elemento con respecto a la línea en la que se encuentra (dentro de su caja contenedora):
   1. block (por defecto para elementos estructurales como div, etc): con display:block; cada elemento ocupa toda la línea (hasta donde llegue su caja contenedora. se utiliza cuando quieras que los blocks se aplien uno debajo de otro
   2. inline (por defecto para elementos de contenido como ‘p’, ‘a’, etc): solo ocupa su espacio.Los elementos inline **no respetan las reglas de tamaño (no se puede indicar height y width)**
   3. inline-block: se ponen las cosas en la misma línea: **su caja ocupa el espacio del elemento, pero sí respeta el width y height.**
2. Position: sirve para especificar cómo se posicionará un elemento (NO SUS CAJAS) con respecto a otro o con la página en sí.
   1. Absolute: hace que se coloque ignorando el resto de elementos.
      1. se puede usar left, top, right, bottom y una unidad (px). Se colocará en esa posición a partir de la línea de la página, ignorando si hay otros elementos (de manera que si hubiera otro se pondría por encima)
   2. Relative: se posiciona en esa posición pero con respecto al elemento anterior (igual que con absolute, se usa left, top etc para indicar la medida de desplazamiento respecto al elemento).
   3. Fixed: el elemento estará posicionado siempre en el lugar de la página en el que está independientemente de que el usuario se desplace con el scroll por la página (estilo para los menús/navigation bars en la cabecera, etc, partes que siempre quieras visible sin importar en qué punto de la página estés).

CENTRAR LOS ELEMENTOS:

Para centrar, se podía usar antes un “margin: 0 auto”, que con los elementos en bloque te los centra.

con CSS3, se añaden unos valores nuevos para las propiedades del display para favorecer la adaptación a mobile.

Posicionamiento avanzado:

1. **Display: flex:** (flex-box) -> es el estándar que se usa ahroa en css par aposicionar las cosas. Hace un grid para ordenar los elementos en un eje (vertical o horizontal, por defecto el horizontal. Posicionar lod elementos de un contenedor en una dimensión.
   1. PAra activarlo, se coge el elemento más alto detodos (el container que tiene los elementos a ordenar.
   2. se indica en su propiedad de display “flex”. los elementos que contenía ahora se disrtibuyen dentro del espacio del padre (modifican su tamaño, ya sea vertical u horizontal).
      1. el padre actúa como un “controlador” para los hijos, que se comportan como dice, en este caso ocupando todo su espacio uniformemente.
   3. Otras propiedades que se pueden usar SOLO si el padre tiene display flex:
      1. flex-direction
         1. row: por defecto. se alinean los hijos horizontalmente
         2. column: se alinean en vertical.
      2. flex-wrap
         1. wrap: los elementos mantienen su tamaño y se colocan uno detrás de otro y los “encaja” en el tamaño del padre. Si no cabe ya horizontalmente, sigue con el resto de hijos en la siguiente línea.
            1. wrap-reverse: se añaden los elementos y cuando se llena la fila, el siguiente se añade en la fila superior (la fila llena quedaría por debajo del nuevo). Se usa en redes sociales, etc.
         2. nowrap: por defecto. ignora el tamaño de los hijos y adapta su tamaño al del padre.
      3. Justify-content: indica cómo utilizan el espacio que sobra en el padre a nivel horizontal:
         1. center: los centra dentro del padre
         2. flex-start: los deja al principio del padre.
         3. flex-end: los coloca a la derecha del padre.
         4. space-between: reparte el espacio sobrante del padre entre los hijos. (sin margen izq/derecho en los elementos de los extremos de cada línea, ni separación vertical)
         5. space-around: reparte el espacio sobrante alrededor de los hijos (tanto lateral izquierdo como derecho de cada elemento)
      4. align-items: lo alinean verticalmente dentro del alto del padre.
         1. center
         2. stretch
         3. flex-start
         4. flex-end

Ejercicio.

colocar 10 elementos con space between, con imágenes, opacity 50% y cuando pases por encima se vuelve opaca.

2. Grid: es como flexbox, pero en las dos direcciones simultaneamnete.

i) se puede mirar el ejemplo de la 23school.

* Grid-template-columns: se hace una lista por cada columna indicando el ancho (se repartirá el espacio en tantas columnas como cifras (px, %, etc) des
  + grid-template-columns: auto auto auto → te repartirá con tamaño automático los elementos que contenga el padre en 3 columnas, dejando espacio con los márgenes del padre por los 4 lados.

**ANIMACIONES**

Una animación en css es una transicion de una propiedad a otra.

para definir una animación, se indica con @keyframes.

con esto le decimos que lo que viene después es el nombre de una animación.

@keyframes change-background{

/\* contenido de la animación\*/

from {

background-color: cyan;

}

to {

background-color: red;

}

}

Para asignar una animación a un elemento, se indica con

animation.name: nombre:animación

animation-duration: tiempo en s/ms para llegar del inicio al final de la animación

animation-iteration-count: cantidad de veces que hará la animación, de 0 a infinite.

**MEDIA QUERIES:**

instrucción que pnes en el código css para hacer que sólo se aplique un estilo si solo se cumple una condición.

Puedes preguntar cosas como si el tamaño de la pantalla tiene una medida X, etc.